□□□□□□□□□□□□□□□□□□□□□□□□□□□□2025v4.
□□□□□□□□□
□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"Neurophysics, Neurochemistry, Neurophysiology and Advanced Artificial Intelligence" 2025v4.3

"Neurophysics, Neurochemistry, Neurophysiology and Advanced Artificial Intelligence" 2025v4.3

Neurophysics, Neurochemistry, Neurophysiology and Advanced Artificial Intelligence 2025v4.3 Global Online E-book

# #

Preface

With the rapid development of artificial intelligence technology, the cross-integration of neuroscience and artificial intelligence has become a key driving force for the development of the next generation of intelligent systems. This book, "Neurophysics, Neurochemistry, Neurophysiology and Advanced Artificial Intelligence" 2025v4.3 Edition, compiles the latest achievements of the world's top research institutions and systematically expounds them.

The following program code and its parameters, in combination with large models, multi-models, and multi-modal general agents, are all written using PATHON,JAVA,C++,MYSQL,MATHLAB, etc., including database code. A. Advanced Neurophysics B. Neurochemistry C. Neurophysiology and Neuromedicine D. Brain Science and Human-Computer Interface, Biological Control E. Genes, cells, physical chemistry, biochemistry, biophysics, F. Superconducting neural reflex arc M of neural networks. Consciousness, memory, association, sleep talking and thinking, reasoning, feedback, control N. Logical thinking, figurative thinking, mathematical logical thinking, natural language thinking, hybrid thinking hybrid

track O. Natural language, mathematical language, figurative language, mixed language and visual, auditory, sensory, tactile and intuitive P. Research and development of chemical and biological drugs for neurological diseases and psychosis, screening and chemical modification of targeted drugs. "Neurophysics, Neurochemistry, Neurophysiology and Advanced Artificial Intelligence" 2025v4.3

Low = NeuroSynthAI - 2.0 "full stack, all disciplines, all die state" = = = = = = = = = = = = = = the overview, target: the unified architecture in complete AP all sub domain modeling, simulation, reasoning, visualization, drugs found. • Technology Stack: - Python 3.11 (PyTorch 2.3, Transformers, Diffusers, CLIP, Whisper, LangChain) - Java 21 (Spring Boot 3.2, Reactor, DL4J) - C++20 (CUDA 12.4, OpenMP, Eigen, Boost, HDF5) - MATLAB R2024a (Simulink, Neural ODE, Symbolic Toolbox) - MySQL 8.3,...

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□2025v4.3 □□□□□□□□□

##

□□

●●●□□□□□□□□□□□□□□□□□□□

utf8mb4_unicode_ci; -- A) □□□□□□ CREATE TABLE neurophysics.simulation_run (id BIGINT AUTO_INCREMENT PRIMARY KEY,model_type ENUM('HH','Izhikevich','AdEx','SC-Circuit') NOT NULL,params JSON, -- □□□□□□□□□□ dt DOUBLE NOT NULL,duration DOUBLE NOT NULL,created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP); -- B) □□□□ CREATE TABLE neurochemistry.reaction (id INT AUTO_INCREMENT PRIMARY KEY,name VARCHAR(255),substrate_smiles TEXT,product_smiles TEXT,km DOUBLE, vmax DOUBLE, temp DOUBLE, ph DOUBLE); -- C) □□□□ CREATE TABLE neuroclinic.patient (id BIGINT PRIMARY KEY,ehr_uuid CHAR(36) UNIQUE,mri_path VARCHAR(512),gene_panel JSON,diagnosis JSON,INDEX idx_uuid (ehr_uuid)); -- E) □□-□□-□□ CREATE TABLE genedock.expression (gene_symbol VARCHAR(30),cell_line VARCHAR(50),tpm FLOAT,condition VARCHAR(100),PRIMARY KEY (gene_symbol, cell_line, condition)); -- P) □□-□□ CREATE TABLE drugbank.compound (id CHAR(7) PRIMARY KEY,smiles TEXT,mw DOUBLE, logp DOUBLE, hba INT, hbd INT, tpsa DOUBLE,toxicity_class ENUM('I','II','III','IV')); -- □□□□□□□□ Lipinski □□ DELIMITER CREATE TRIGGER trg_lipinski BEFORE INSERT ON drugbank.compoundFOR EACH ROWBEGINIF NEW.mw > 500 OR NEW.logp > 5 OR NEW.hba > 10 OR NEW.hbd > 5 THENSIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lipinski violation';END IF;ENDDELIMITER ; ---3 □□□□□□□□□□ 3.1 A. □□□□□□ — Python□□□ HH □□□ ```python# file: neurophysics/sc_hh.pyimport numpy as npfrom scipy.integrate import solve_ivpimport torchclass SuperconductingHH(torch.nn.Module): def __init__(self, C=1.0, gNa=120, gK=36, gL=0.3, ENa=50, EK=-77, EL=-54.4): super().__init__() self.register_buffer('params', torch.tensor([C,gNa,gK,gL,ENa,EK,EL])) def forward(self, t, y): V,m,h,n = y C,gNa,gK,gL,ENa,EK,EL = self.params I_ext = 10*(t>10)*(t<80) # μA/cm² dV = (I_ext - gNa*m**3*h*(V-ENa) - gK*n**4*(V-EK) - gL*(V-EL)) / C # □□□□□ return torch.stack([dV, dm, dh, dn])```3.2 B. □□□□ — Java□□□□□□ ```java// file: src/main/java/neurochem/Reaction.javapublic record Reaction(String substrate, String product, double km, double vmax, double temp, double ph) { public double rate(double s) { return vmax * s / (km + s) * Math.pow(10, ph-7); }}```3.3 C. □□□□□□□□ — C++□□□ fMRI □□□ ```cpp// file: src/cpp/rt_fmri.cpp#include <opencv2/opencv.hpp>#include <torch/script.h>class RealTimeDecoder { torch::jit::script::module model;public: RealTimeDecoder(const std::string& path) { model = torch::jit::load(path); } cv::Mat decode(const cv::Mat& vol) { torch::Tensor t = torch::from_blob(vol.data, {1,64,64,32}, torch::kFloat32); auto out = model.forward({t}).toTensor(); return cv::Mat(64,64,CV_32F,out.data_ptr<float>()); }};```3.4 D. □□□□ — Python ROS2 □□ ```python# file: bci/spike_decoder.pyimport rclpyfrom std_msgs.msg import Float32MultiArrayimport torchclass SpikeDecoderNode(rclpy.node.Node): def __init__(self): super().__init__('spike_decoder') self.create_subscription(Float32MultiArray, '/raw_spikes', self.cb, 10) self.model = torch.jit.load('/models/spike_cnn.pt') def cb(self, msg): x = torch.tensor(msg.data).reshape(1,1,128) y = self.model(x).squeeze().argmax().item() self.get_logger().info(f'Decoded class {y}')```3.5 E. □□-□□-□□ — MATLAB ```matlab% file: genedock/ode_gene_expression.mfunction dydt = ode_gene_expression(~,y,p) %

y = [mRNA, Protein] dydt = zeros(2,1); dydt(1) = p.alpha - p.beta*y(1); dydt(2) = p.k*y(1) - p.gamma*y(2);end```3.6 F. □□□□□□ — C++ + CUDA ```cpp// file: src/cuda/super_neuron.cu__global__ void sc_update(float* V, float* I, int N, float g_leak, float C, float dt) { int i = blockIdx.x * blockDim.x + threadIdx.x; if (i < N) { V[i] += dt * (-g_leak * (V[i] + 70) + I[i]) / C; }}```3.7 M. □□-□□-□□ — Python Transformer □□□□ ```python# file: memory/dreamer.pyfrom transformers import GPT2LMHeadModel, GPT2Tokenizertok = GPT2Tokenizer.from_pretrained('gpt2')model = GPT2LMHeadModel.from_pretrained('gpt2')prompt = "I was walking in a dark forest when suddenly"ids = tok.encode(prompt, return_tensors='pt')out = model.generate(ids, max_length=100, do_sample=True, top_p=0.9)print(tok.decode(out[0]))```3.8 N. □□□□□ — Java Spring Boot □□□□ ```java@RestController@RequestMapping("/reason")public class ReasonController { @PostMapping("/hybrid") public Mono<ResponseEntity<JsonNode>> hybridThink(@RequestBody JsonNode input) { return reactiveModel.infer(input) .map(r -> ResponseEntity.ok(r)); }}```3.9 O. □□□□□ — Python FastAPI ```pythonfrom fastapi import FastAPI, File, UploadFileimport clip, whisper, cv2, torchapp = FastAPI()model, _ = clip.load("ViT-B/32", device="cuda")@app.post("/multimodal")async def multimodal(text: str = None, audio: UploadFile = None, image: UploadFile = None): feats = [] if text: feats.append(clip.tokenize(text).cuda()) if image: img = cv2.imdecode(np.frombuffer(await image.read(), np.uint8), 1) feats.append(model.encode_image(preprocess(img).unsqueeze(0).cuda())) # □□□□□```3.10 P. □□□□ — Python + RDKit + MySQL ```python# file: drugai/screen.pyfrom rdkit import Chem, DataStructsfrom rdkit.Chem import AllChemimport mysql.connector as sqlconn = sql.connect(host='mysql', user='root', password='neuro', database='drugbank')cur = conn.cursor()cur.execute("SELECT id, smiles FROM compound WHERE toxicity_class='III'")for cid, smi in cur: m = Chem.MolFromSmiles(smi) fp = AllChem.GetMorganFingerprintAsBitVect(m, 2, nBits=2048) # □□□□□□□□□□```---4 □ □□□□Docker Compose□ ```yaml# file: docker-compose.prod.ymlversion: "3.9"services: mysql: image: mysql:8.3 environment: MYSQL_ROOT_PASSWORD: neuro ports: ["3306:3306"] neo4j: image: neo4j:5.15 environment: NEO4J_AUTH: neo4j/neuro api: build: ./backend depends_on: [mysql, neo4j] ports: ["8000:8000"] matlab: image: mathworks/matlab:r2024a volumes: ["./matlab:/code"] command: matlab -batch "run('/code/main.m')"```□□□ ```bashdocker compose -f docker-compose.prod.yml up --build```---5 □□□ K8s□GPU □□□ ```bashhelm install neurosynth ./helm/neurosynth \ --set gpu.enabled=true \ --set mysql.storage=100Gi \ --set ingress.host=ai.neuro.gov```

●□□□□□□□□□

__init__(self, id, threshold): self.id = id self.threshold = threshold self.potential = 0 def receive_signal(self, signal): self.potential += signal if self.potential >= self.threshold: self.potential = 0 return 1 # □□□□ return 0 □□□□□□□□□□ neuron1 = NeuronAgent(1, 10) neuron2 = NeuronAgent(2, 10) □□□□□□ signals = np.random.randint(1, 5, 10) for signal in signals: output = neuron1.receive_signal(signal) if output: neuron2.receive_signal(output) print(f"Neuron {neuron1.id} fired and sent signal to Neuron {neuron2.id}``` Java □□ Java □□□ ```java class NeuronAgent { private int id; private double threshold; private double potential; public NeuronAgent(int id, double threshold) { this.id = id; this.threshold = threshold; this.potential = 0; } public int receiveSignal(double signal) { potential += signal; if (potential >= threshold) { potential = 0; return 1; // □□□□ } return 0; } public static void main(String[] args) { NeuronAgent neuron1 = new NeuronAgent(1, 10); NeuronAgent neuron2 = new NeuronAgent(2, 10); for (int i = 0; i < 10; i++) { int randomSignal = (int) (Math.random() * 4 + 1); int output = neuron1.receiveSignal(randomSignal); if (output == 1) { neuron2.receiveSignal(output); System.out.println("Neuron " + neuron1.id + " fired and sent signal to Neuron " + neuron2.id); } } } } ``` C++□□ C++□□□□□□□□ □□□□□□□□□□□□□□□□□□□□ ```cpp #include <iostream> #include <cstdlib> #include <ctime> class NeuronAgent { private: int id; double threshold; double potential; public: NeuronAgent(int id, double threshold) : id(id), threshold(threshold), potential(0) {} int receiveSignal(double signal) { potential += signal; if (potential >= threshold) { potential = 0; return 1; // □□□□ } return 0; } }; int main() { NeuronAgent neuron1(1, 10); NeuronAgent neuron2(2, 10); srand(time(NULL)); for (int i = 0; i < 10; i++) { int randomSignal = rand() % 4 + 1; int output = neuron1.receiveSignal(randomSignal); if (output == 1) { neuron2.receiveSignal(output); std::cout << "Neuron " << neuron1.id << " fired and sent signal to Neuron " << neuron2.id << std::endl; } } return 0; } ```
●●□□ □□□□□□□□□□□□□□□□□□□□

for neuron in layer: layer_outputs.append(neuron.forward(activations)) activations = np.array(layer_outputs) return activations def train(self, X, y, epochs=1000): """□□□□□□""" losses = [] for epoch in range(epochs): epoch_loss = 0 for i in range(len(X)): # 1. □□□□ output = self.forward(X[i]) # 2. □□□□□□□□□□□ loss = np.mean((output - y[i]) ** 2) epoch_loss += loss # 3. □□□□□□□□□□ # □□□□□□□ BPTT □□ error = output - y[i] # □□□□□□□□□□□□□□□□ for layer in reversed(self.layers): for neuron in layer: # □□□□□□□□□□□□ neuron.weights -= self.lr * error * neuron.last_activation neuron.bias -= self.lr * error losses.append(epoch_loss/len(X)) if epoch % 100 == 0: print(f"Epoch {epoch}, Loss: {losses[-1]:.4f}") # □□□□□□□□ plt.plot(losses) plt.title("Training Loss Curve") plt.xlabel("Epochs") plt.ylabel("Loss") plt.show()□□□□ if __name__ == "__main__": # 1. □□□□□□□□□□□□ X = np.array([[0,0], [0,1], [1,0], [1,1]]) y = np.array([[0], [1], [1], [0]]) # 2. □□□□□ [□□□ 2 □□, □□□ 4 □□, □□□ 1 □□] nn = NeuralNetwork(layers=[2, 4, 1], lr=0.1) # 3. □□□□ nn.train(X, y, epochs=2000) # 4. □□□□ print("Predictions:") for sample in X: print(f"{sample} => {nn.forward(sample)[0]:.2f}")```□□□□□□ 1. □□□□□□ - □□ LSTM/GRU □□□□□□□□ ```python class LSTMCell(Neuron): # □□□□□□□□□□ def __init__(self, n_inputs): super().__init__(n_inputs + n_inputs) # □□+□□□□ # □□□□□/□□□/□□□ ```2. □□□□ - □□ □□□□□□□ BPTT ```python def backpropagate(self, error): # □□□□□□□ deltas = [error * self._activation_deriv(output)] for i in reversed(range(len(self.layers)-1)): deltas.append(deltas[-1].dot(self.weights[i].T) * self._activation_deriv(self.activations[i])) ```3. □□□□□ - □□□□□□□□ ```python class ConvLayer: # □□□□□□ def __init__(self, filters, kernel_size): self.filters = [np.random.randn(*kernel_size) for _ in range(filters)] ```4. □□□□ - □□ Dropout □□□ ```python def forward(self, inputs, training=False): if training: self.mask = (np.random.rand(*inputs.shape) > dropout_rate) inputs *= self.mask ```□□□□□□ | □□ | □□□ | □□ ||---------------|-------------|-------------------------|| layers | [2,4,1] | □□□□□ || lr | 0.01-0.1 | □□□□□□□□□ || activation | sigmoid/relu| □□□□□ relu || epochs | 1000-5000 | □□□□ |> □□□□□□□ Keras □□ API □□ RNN □□□□□□ BP □□□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□- □□`save_model()`/`load_model()`□□□- □□□□□□/□□□□□□ - □□ TensorFlow □ □□□□□- □□□□□□□□□□□□□□□

●□□□□□□□□□□□□□□□

"CC(=O)OC1=CC=CC=C1C(=O)O"])3. □□□□□□□MATLAB□% □□ EEG □□□□ fs = 250; % □□□ data = readmatrix('eeg_data.csv'); epochs = segmentData(data, fs, 0.5, 0.1); % 0.5-0.1 □□□□□% □□□□□ CNN □□ net = load('eeg_cnn.mat');predicted_labels = classify(net, epochs);disp(['Detected mental state: ', char(predicted_labels)]);□□ □□□□□□MySQL□-- □□□□□□□□□ CREATE DATABASE neuro_science;USE neuro_science;-- □□□□ CREATE TABLE drugs (id INT AUTO_INCREMENT PRIMARY KEY, smiles VARCHAR(255) NOT NULL, molecular_weight FLOAT, blood_brain_barrier BOOLEAN);-- □□□□□□ CREATE TABLE eeg_data (session_id VARCHAR(36), timestamp DATETIME, channel1 FLOAT, channel2 FLOAT, INDEX idx_session (session_id));-- □□□□□ CREATE TABLE diseases (disease_id INT PRIMARY KEY, name VARCHAR(100), symptoms TEXT, linked_genes VARCHAR(255));□□□□□□□□□# □□□□□□□□PyTorch□import torchfrom transformers import BertModelfrom torchvision.models import resnet50class NeuroMultimodal(torch.nn.Module): def __init__(self): super().__init__() self.text_encoder = BertModel.from_pretrained('bert-base-uncased') self.image_encoder = resnet50(pretrained=True) self.fusion = torch.nn.Linear(768+2048, 512) def forward(self, text, image): text_feat = self.text_encoder(**text).last_hidden_state img_feat = self.image_encoder(image) fused = torch.cat([text_feat, img_feat], dim=1) return self.fusion(fused)□□□□□□□□ 1. □□□□MySQL □□□□□□□□□□MongoDB □□□□□□□□ 2. □□□□ - Python□□□□□□□□□□PyTorch/TensorFlow□ - C++□□□□□□□□OpenCV/FFTW□ - Java□□□□ API □□□Spring Boot□3. □□□□MATLAB □□□□ + Unity3D □□□□□□□□□□□□□□□□ □□□□ □□ □□□□□□□□□ □□□□□□ □□ LSTM □□□□□□□□□□□□ □□□□□□ □□□□□□□□□DFT□□□□□□ □ □□□□□□ FPGA □□□□□□□□□□□□□ 1. □□□□-□□□□□# □□□□□□□□OpenMM□from openmm import app, unitsystem = app.ForceField('amber14-all.xml').createSystem(mol.topology)integrator = openmm.LangevinIntegrator(300*unit.kelvin, 1/unit.picosecond, 0.002*unit.picoseconds)simulation = app.Simulation(mol.topology, system, integrator)simulation.step(1000000) # 100ns □□ 2. □□□-□□□□□// □□□□□□ □TensorFlow Lite□#include <tensorflow/lite/interpreter.h>TfLiteInterpreter* interpreter = new TfLiteInterpreter(model, nullptr);interpreter->AllocateTensors();float input[256] = {eeg_samples};interpreter->tensor(inputIndex)->data.f = input;interpreter->Invoke();float* output = interpreter->tensor(outputIndex)->data.f;□□□□□□□□ 1. □□□□□# □□□□□□□□□□□ python test_eeg.py --dataset=BCI_IV_IIa --model=resnet182. □□□□□□□# □□□□□□□ □ start_time = time.time()screened = screen_drugs(database['molecules'])print(f"□□□□: {time.time()-start_time}□")□□□ □□□□□□□□□□□□□□□□

neurotransmitter_type VARCHAR(50), location_brain_region VARCHAR(100), electrical_properties JSON, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);-- □□□□□□□ CREATE TABLE Neurochemicals (chemical_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, formula VARCHAR(50), molecular_weight DECIMAL(10,4), function_description TEXT, receptor_types JSON, synthesis_pathway TEXT);-- □□□□□ CREATE TABLE BrainRegions (region_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, location_description TEXT, primary_function TEXT, connected_regions JSON, neurotransmitter_profile JSON);-- □□□□□ CREATE TABLE NeurologicalDisorders (disorder_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, description TEXT, affected_brain_regions JSON, biochemical_changes TEXT, treatment_options JSON);-- □□□□□ CREATE TABLE DrugDevelopment (drug_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, target_neurotransmitter VARCHAR(100), mechanism_of_action TEXT, chemical_structure BLOB, trial_results JSON, side_effects TEXT);-- □□□□□□□ CREATE TABLE CognitiveExperiments (experiment_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(200) NOT NULL, description TEXT, participant_data JSON, neural_activity_records LONGBLOB, analysis_results JSON, conducted_at TIMESTAMP);```## Python □□□□□□### 1. □□□□□□□□□□```pythonimport numpy as npimport matplotlib.pyplot as pltfrom scipy.integrate import odeintclass HodgkinHuxleyModel:

●□□“□□□□□□□□□□□□”□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ P/E/O□□□□□□□□□□□□□□□□

<cmath>#include <fstream>// □□□□□□□□□□□□□□□class BCIProcessor {private: const int SAMPLE_RATE = 1000; // □□□□Hz□ const int WINDOW_SIZE = 256; // □□□□□□ std::vector<double> signal_buffer; // □□□□□ public: // □□□□□□ double process_frame(double raw_signal) { signal_buffer.push_back(raw_signal); if (signal_buffer.size() > WINDOW_SIZE)
{ signal_buffer.erase(signal_buffer.begin()); // □□□□ } return extract_feature(); // □□□□□□□□□□□□ } // □□□□□□□□□□□□□□ β □□□□ double extract_feature() { if (signal_buffer.size() < WINDOW_SIZE) return 0.0; double mean = 0.0, var = 0.0; for (double s : signal_buffer) mean += s; mean /= WINDOW_SIZE; for (double s : signal_buffer) var += pow(s - mean, 2); var /= WINDOW_SIZE; return var; // □□□□ □□□□ } // □□□□□□□□□□□□□□□□□□□ std::string decode_intent(double feature) { return (feature > 0.5) ? "□□□□□□□□□" : "□□□□□"; }};int main() { BCIProcessor bci; std::ifstream signal_file("real_time_eeg.txt"); // □□□□□□□□ double signal; // □□□□□□ □ 1000 □□ while (signal_file >> signal) { double feat = bci.process_frame(signal); std::string intent = bci.decode_intent(feat); std::cout << "□□□□□" << feat << "□ □□□□□" << intent << std::endl; } return 0;} 4. Java□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ API □□□□□□□□□□□□□□□□import java.sql.*;import com.google.gson.*; // □□ JSONpublic class NeuroService { // □□□□□ private Connection getConnection() throws SQLException { return DriverManager.getConnection("jdbc:mysql://localhost:3306/neuro_science", "root", ""); } // □□□□□□ public String getTargetDrugs(String disease) throws SQLException { Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement("SELECT drug_name, target_protein, screening_score FROM target_drugs " + "JOIN neurological_diseases ON target_drugs.disease_id = neurological_diseases.disease_id " + "WHERE disease_name LIKE ?"); stmt.setString(1, "%" + disease + "%"); ResultSet rs = stmt.executeQuery(); JsonArray drugs = new JsonArray(); while (rs.next()) { JsonObject drug = new JsonObject(); drug.addProperty("name", rs.getString("drug_name")); drug.addProperty("target", rs.getString("target_protein")); drug.addProperty("score", rs.getDouble("screening_score")); drugs.add(drug); } conn.close(); return new Gson().toJson(drugs); } // □□□□□□□□□□□□ Python □□□□□□□ public String runDrugScreening(String drugFormula) { // □□□□□ JNI □ HTTP □□ Python □□ return "{\"status\":\"success\", \"message\":\"□□□□□□\"}"; } public static void main(String[] args) throws SQLException { NeuroService service = new NeuroService(); String result = service.getTargetDrugs("□□□□□□"); System.out.println("□□□□□□□" + result); }} □□□□□□□□ 1. □□□□□□□□□250-1000Hz□□ □□□□□α □ 8-12Hz□β □ 13-30Hz□□□□□□□128-512 □□□□2. □□□□□□□□□□□□□□□□logP□□□□□ □□□□□□□□□□□□>0.6 □□□□□□□□□3. □□□□□□□□□ BERT □□□□□□□□□ 768□□□□□ ResNet50 □□□□□ 2048□□4. □□□□□□□□□ BLOB □□□□□□□□□□□□□□JSON□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□1. □□□□□□□□□□□□□□□□□□□□□□□□ EEG □□□□□2. □□□□□□□□□□□□□□ RDKit□□□□□□ MNE-Python□□3. □□□□□□□□□□□□□□□□□□□□□□□□ LLM□□4. □□□□□□□□□□□□□□□□□□□□ CLIP □□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□

●

□□□□ • □□□□□□□□□□□□ AP □□□□□□□□□□□□□□□□□□□□□□□□□□□□□• □□□□- Python 3.11□PyTorch 2.3□Transformers□Diffusers□CLIP□Whisper□LangChain□- Java 21□Spring Boot

3.2□Reactor□DL4J□- C++20□CUDA 12.4□OpenMP□Eigen□Boost□HDF5□- MATLAB R2024a□Simulink□Neural ODE□Symbolic Toolbox□- MySQL 8.3□Neo4j 5.x□MongoDB 7.x□Redis 7.x- ROS2 Humble + DDS + ZeroMQ□□□□□□□□□- Kubernetes 1.29 + Helm 3.14 + GPU Operator ---2 □□□□□□□MySQL 8.3□ -- 2.1 □□□□ SET GLOBAL innodb_buffer_pool_size = 32G;SET GLOBAL sql_mode = 'STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO'; -- 2.2 □□ ER □□□□ □□CREATE SCHEMA neurosynth CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; -- A) □□□□□□ CREATE TABLE neurophysics.simulation_run (id BIGINT AUTO_INCREMENT PRIMARY KEY,model_type ENUM('HH','Izhikevich','AdEx','SC-Circuit') NOT NULL,params JSON, -- □□□□□□□□□□ dt DOUBLE NOT NULL,duration DOUBLE NOT NULL,created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP); -- B) □□□□ CREATE TABLE neurochemistry.reaction (id INT AUTO_INCREMENT PRIMARY KEY,name VARCHAR(255),substrate_smiles TEXT,product_smiles TEXT,km DOUBLE, vmax DOUBLE, temp DOUBLE, ph DOUBLE); -- C) □□□□ CREATE TABLE neuroclinic.patient (id BIGINT PRIMARY KEY,ehr_uuid CHAR(36) UNIQUE,mri_path VARCHAR(512),gene_panel JSON,diagnosis JSON,INDEX idx_uuid (ehr_uuid)); -- E) □□-□□-□□ CREATE TABLE genedock.expression (gene_symbol VARCHAR(30),cell_line VARCHAR(50),tpm FLOAT,condition VARCHAR(100),PRIMARY KEY (gene_symbol, cell_line, condition)); -- P) □□-□□ CREATE TABLE drugbank.compound (id CHAR(7) PRIMARY KEY,smiles TEXT,mw DOUBLE, logp DOUBLE, hba INT, hbd INT, tpsa DOUBLE,toxicity_class ENUM('I','II','III','IV')); -- □□□□□□□□ Lipinski □□ DELIMITER CREATE TRIGGER trg_lipinski BEFORE INSERT ON drugbank.compoundFOR EACH ROWBEGINIF NEW.mw > 500 OR NEW.logp > 5 OR NEW.hba > 10 OR NEW.hbd > 5 THENSIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lipinski violation';END IF;ENDDELIMITER ; ---3 □□□□□□□□□□ 3.1 A. □□□□□□ — Python□□□ HH □□□ ```python# file: neurophysics/sc_hh.pyimport numpy as npfrom scipy.integrate import solve_ivpimport torchclass SuperconductingHH(torch.nn.Module): def __init__(self, C=1.0, gNa=120, gK=36, gL=0.3, ENa=50, EK=-77, EL=-54.4): super().__init__() self.register_buffer('params', torch.tensor([C,gNa,gK,gL,ENa,EK,EL])) def forward(self, t, y): V,m,h,n = y C,gNa,gK,gL,ENa,EK,EL = self.params I_ext = 10*(t>10)*(t<80) # μA/cm² dV = (I_ext - gNa*m**3*h*(V-ENa) - gK*n**4*(V-EK) - gL*(V-EL)) / C # □□□□□ return torch.stack([dV, dm, dh, dn])```3.2 B. □□□□ — Java□□□□□□ ```java// file: src/main/java/neurochem/Reaction.javapublic record Reaction(String substrate, String product, double km, double vmax, double temp, double ph) { public double rate(double s) { return vmax * s / (km + s) * Math.pow(10, ph-7); }}```3.3 C. □□□□□□□□ — C++□□□ fMRI □□□ ```cpp// file: src/cpp/rt_fmri.cpp#include <opencv2/opencv.hpp>#include <torch/script.h>class RealTimeDecoder { torch::jit::script::module model;public: RealTimeDecoder(const std::string& path) { model = torch::jit::load(path); } cv::Mat decode(const cv::Mat& vol) { torch::Tensor t = torch::from_blob(vol.data, {1,64,64,32}, torch::kFloat32); auto out = model.forward({t}).toTensor(); return cv::Mat(64,64,CV_32F,out.data_ptr<float>()); }};```3.4 D. □□□□ — Python ROS2 □□ ```python# file: bci/spike_decoder.pyimport rclpyfrom std_msgs.msg import Float32MultiArrayimport torchclass SpikeDecoderNode(rclpy.node.Node): def

__init__(self): super().__init__('spike_decoder') self.create_subscription(Float32MultiArray, '/raw_spikes', self.cb, 10) self.model = torch.jit.load('/models/spike_cnn.pt') def cb(self, msg): x = torch.tensor(msg.data).reshape(1,1,128) y = self.model(x).squeeze().argmax().item() self.get_logger().info(f'Decoded class {y}')```3.5 E. □□-□□-□□ — MATLAB ```matlab% file: genedock/ode_gene_expression.mfunction dydt = ode_gene_expression(~,y,p) % y = [mRNA, Protein] dydt = zeros(2,1); dydt(1) = p.alpha - p.beta*y(1); dydt(2) = p.k*y(1) - p.gamma*y(2);end```3.6 F. □□□□□□ — C++ + CUDA ```cpp// file: src/cuda/super_neuron.cu__global__ void sc_update(float* V, float* I, int N, float g_leak, float C, float dt) { int i = blockIdx.x * blockDim.x + threadIdx.x; if (i < N) { V[i] += dt * (-g_leak * (V[i] + 70) + I[i]) / C; }}```3.7 M. □□-□□-□□ — Python Transformer □□□□ ```python# file: memory/dreamer.pyfrom transformers import GPT2LMHeadModel, GPT2Tokenizertok = GPT2Tokenizer.from_pretrained('gpt2')model = GPT2LMHeadModel.from_pretrained('gpt2')prompt = "I was walking in a dark forest when suddenly"ids = tok.encode(prompt, return_tensors='pt')out = model.generate(ids, max_length=100, do_sample=True, top_p=0.9)print(tok.decode(out[0]))```3.8 N. □□□□□ — Java Spring Boot □□□□ ```java@RestController@RequestMapping("/reason")public class ReasonController { @PostMapping("/hybrid") public Mono<ResponseEntity<JsonNode>> hybridThink(@RequestBody JsonNode input) { return reactiveModel.infer(input) .map(r -> ResponseEntity.ok(r)); }}```3.9 O. □□□□□ — Python FastAPI ```pythonfrom fastapi import FastAPI, File, UploadFileimport clip, whisper, cv2, torchapp = FastAPI()model, _ = clip.load("ViT-B/32", device="cuda")@app.post("/multimodal")async def multimodal(text: str = None, audio: UploadFile = None, image: UploadFile = None): feats = [] if text: feats.append(clip.tokenize(text).cuda()) if image: img = cv2.imdecode(np.frombuffer(await image.read(), np.uint8), 1) feats.append(model.encode_image(preprocess(img).unsqueeze(0).cuda())) # □□□ □□```3.10 P. □□□□ — Python + RDKit + MySQL ```python# file: drugai/screen.pyfrom rdkit import Chem, DataStructsfrom rdkit.Chem import AllChemimport mysql.connector as sqlconn = sql.connect(host='mysql', user='root', password='neuro', database='drugbank')cur = conn.cursor()cur.execute("SELECT id, smiles FROM compound WHERE toxicity_class='III'")for cid, smi in cur: m = Chem.MolFromSmiles(smi) fp = AllChem.GetMorganFingerprintAsBitVect(m, 2, nBits=2048) # □□□□□□□□□□```---4 □ □□□□Docker Compose□ ```yaml# file: docker-compose.prod.ymlversion: "3.9"services: mysql: image: mysql:8.3 environment: MYSQL_ROOT_PASSWORD: neuro ports: ["3306:3306"] neo4j: image: neo4j:5.15 environment: NEO4J_AUTH: neo4j/neuro api: build: ./backend depends_on: [mysql, neo4j] ports: ["8000:8000"] matlab: image: mathworks/matlab:r2024a volumes: ["./matlab:/code"] command: matlab -batch "run('/code/main.m')"```□□□ ```bashdocker compose -f docker-compose.prod.yml up --build```---5 □□□ K8s□GPU □□□ ```bashhelm install neurosynth ./helm/neurosynth \ --set gpu.enabled=true \ --set mysql.storage=100Gi \ --set

ingress.host=ai.neuro.gov```---6 □□□ • □□□□ GitHub □□□□ .git□CI/CD□□□□□□□□□□“□□□□□ zip”• □□□□□□□□HCP fMRI□Allen Cell□ChEMBL□□□□□□“□□□□□□”

●●□□□□□□□□□□

1. □□□□Conv2D□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- PyTorch □□□

```python

import torch.nn as nn

conv_layer = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)

```

- □□□□□□`in_channels`□□□□□□□RGB □□□ 3□□`out_channels`□□□□□□□□□`kernel_size`□□□□□

2. □□□□ReLU□

- □□□□□□□`nn.ReLU()` □ `F.relu()`

- □□□□□□□□□□□□□

3. □□□□Pooling□

- □□□□□□`nn.MaxPool2d(kernel_size=2, stride=2)`

- □□□□□□□□□□□□□□

4. □□□□□□□BatchNorm□

- □□□□□□`nn.BatchNorm2d(num_features=16)`

- □□□□□□□□□□□□□

□□□□□□□□□□□□□□□

```python

import torch.nn as nn
```